Appendix E

xSides Corporation
821 Second Avenue, #1600
Seattle, WA 98104

xSides Corporation

xSides™ Video Driver Extensions

Draft 0.2

xSides™ Video Driver Extensions

Overview

Introduction

This document describes the video driver extensions necessary to provide a generic interface between the xSides application and the Windows video driver. These extensions allow the display of xSides pixel bars in the video overscan region that exists at the edges of the video display, outside the Windows Desktop. The display of the xSides pixel bars, specifically the number of pixel bars or location of the pixel bars - -Top, Bottom, Right, or Left edge -- does not reduce the pixel resolution of the Windows desktop.

To implement the overscan regions, a set of video driver extensions are defined and implemented via driver escapes. The xSides application, which is a Win32 application running in user mode, creates, modifies, deletes and draws in the pixel bars by calling the GDI (Graphics Display Interface) Escape function, and passing it messages that the video driver understands. These messages are called the xSides Video Driver Extensions.

Support for the mouse cursor in the pixel bars is also defined to provide the xSides application the capability to define a cursor icon, and to position it in the pixel bars beyond the normal Windows Desktop.

Three levels of support are defined:

1. Level One supports only the bottom pixel bar.
2. Level Two supports both the bottom and the top pixel bars.
3. Level Three supports the bottom, top, left and right pixel bars.

Level One Support

Video drivers that provide xSides level one support allow the display of pixel bar (aka *Side*) at the bottom of the display, beyond the Windows desktop. The video driver can provide this capability by allocating video memory contiguous to the frame buffer, increasing the number of visible scanlines, and having the capability to BitBlt bitmaps to the allocated video memory.

Level Two Support

Video drivers that provide xSides level two support allow the display of pixel bars (aka *Sides*) at the top and the bottom of the display, above and below the Windows desktop respectively. The video driver can provide this capability by allocating enough video memory contiguous to the frame buffer for both pixel bars (aka Sides), increasing the number of visible scanlines sufficent for both pixel bars, the having capability to BitBlt bitmaps to either the top or bottom pixel bar, and offsetting the Windows Desktop downwards in the framebuffer so that the very top of the video framebuffer is taken up by the top pixel bar.

Level Three Support

Video drivers that provide xSides with level three support allow the display of pixel bars (aka *Sides*) at the left and right sides of the display, to the left and to the right, respectively, of the Windows desktop, in addition to the display of pixel bars at the top and bottom of the display. To provide this support, in addition to the capabilities described for level two support, the frame buffer could be organized so that the screen stride is larger than the Windows desktop horizontal byte count, to provide video memory for the left and right pixel bars, and the horizontal display resolution would need to be increased to make the pixel bars visible.

Mouse Cursor (Pointer) Support

Video drivers that provide xSides with mouse cursor support allow the definition and display of the cursor icon in the pixel bar screen regions, beyond the Windows desktop. To provide the support, the video driver must maintain a separate cursor icon cache for xSides, and use it when a request is made to display the mouse cursor in the xSides pixel bar(s). When a pixel bar is active along any edge of the desktop, the cursor icon should not be clipped at the desktop boundary, but rather should be extended onto the pixel bar until the cursor hotspot reaches the edge of the desktop. Once the cursor hotspot transitions out of the Windows Desktop, the normal cursor icon should be deactivated and made invisible.

Configuration Information

In order for the xSides application to size pixel bars appropriately, the video driver must return configuration information regarding the maximum amount of extra screen space that can be supported for each resolution . This is determined primarily by the amount of video memory and video bandwidth available. The xSides application will use the configuration information from the video driver, based upon the video adapter and monitor characteristics, and determine the optimum sizes for the pixel bars while minimizing refresh frequency degradation.

DirectDraw Support for xSides

There is interest in having the display driver support the assigning of DirectDraw surface handles to the pixel bars. The details are TBD.

Display Events

There is interest in having the display driver supporting the report of display events to the xSides application. These events would include:

- DirectDraw Full Screen start.
- DirectDraw Full Screen end.
- Screen Saver start.
- Screen Saver end.
- Dos Full Screen start.
- Dos Full Screen end.
- Message Mode (aka Blue Screen of Death) start.
- Message Mode (aka Blue Screen of Death) end.
- Display Resolution Change.
- Display Driver errors, especially in the xSides support code.

It's desired that the events be reported asynchronously, probably by callback functions. The details are TBD.

xSides Driver Escapes

The Video Driver must support an xSides Escape code (defined as ESC_xSIDES; the numerical value of the code is TBD), and report that it is being supported when queried with a QUERYESCSUPPORT Escape code. In Windows NT, the GDI ExtEscape function will be used by the xSides application.

The Windows GDI escape functions are described in Appendix A.

xSides Escapes

All the xSides escapes use ESC_xSIDES for the nEscape parameter to the GDI function ExtEscape. The lpszInData parameter points to a structure that

contains the specific xSides message. The video driver returns output messages back to the xSides application in the buffer pointed to by the lpszOutData parameter. The following xSides messages are defined to support the display of the pixel bars:

- **IMSG_XSIDES_BEGIN** notifies the video driver that the xSides application has started executing and requires video driver support. The video driver returns the **OMSG_XSIDES_STATUS** message in response, with the appropriate status and error information.

- **IMSG_XSIDES_END** notifies the video driver that the xSides application is exiting and no longer requires video driver support. The video driver returns the **OMSG_XSIDES_STATUS** message in response, with the appropriate status and error information.

- **IMSG_XSIDES_QUERY_VIDEO_INFO** queries the video driver for information about the video hardware and driver. Among the information requested are the version of the xSides video driver extensions and the level of functionality that the video driver supports. The video driver returns the **OMSG_XSIDES_VIDEO_INFO** message.

- **IMSG_XSIDES_CREATE_PIXELBAR** commands the video driver to create the specified pixel bar -- bottom, top, left, or right. This includes allocating video memory for the pixel bar, reprogramming the video hardware to increase the vertical or horizontal pixel resolution, offsetting the Windows desktop (in the case of the top and left bars), and changing the screen stride (in the case of the left and right pixel bars.) The video driver returns the **OMSG_XSIDES_STATUS** message in response, with the appropriate status and error information.

- **IMSG_XSIDES_DELETE_PIXELBAR** commands the video driver to delete a previously created pixel bar. This may include deallocating video memory, reprogramming the video hardware, removing offsets from the display of the Windows desktop and changing the screen stride. The video driver returns the **OMSG_XSIDES_STATUS** message in response, with the appropriate status and error information.

- **IMSG_XSIDES_DISPLAY_PIXELBAR** commands the video driver to do a BitBlt of the specified bitmap to a previously created pixel bar at the specified position with the specified extents. The source bitmap is in the current screen format. For this message, the lpszOutData parameter points to the source bitmap.

- **IMSG_XSIDES_ACTIVATE_PIXELBAR** commands the video driver to make a pixel bar visible by increasing the number of pixels displayed on the display surface.

- **IMSG_XSIDES_DEACTIVATE_PIXELBAR** commands the video driver to do a.
- **IMSG_XSIDES_ENABLE_CURSOR** commands the video driver to display the mouse cursor icon in the pixel bar.
- **IMSG_XSIDES_DISABLE_CURSOR** commands the video driver to not display the mouse cursor icon in the pixel bar.
- **IMSG_XSIDES_SET_CURSOR** downloads the xSides mouse cursor icon to the video driver.
- **IMSG_XSIDES_MOVE_CURSOR** commands the video driver to move the xSides mouse cursor to a new location in the pixel bar.

xSides Escape Messages

Generic Message Structure

The xSides messages begin with a message code and size fields, followed by message specific data fields. The message code indicates the specific message; the size indicates the number of bytes in the message, including the code and size fields. In the figures below, Little Endian byte ordering (where the most significant byte of a multi-byte value has the highest address) is assumed, with the most significant byte at the left and the least significant byte at the right.

The generic xSides Escape message structure is as follows:

USHORT Code	USHORT Size
[Others -- Message Specific fields]	

Code: The message code, in bytes 2 and 3 of the message structure.

Size: The size of the message, including the Code and Size fields, in bytes 0 and 1 of the message structure.

Others: Message specific fields whose number, format and definition vary from message to message. Some messages may have none.

The structure would be defined in C as:

```
typedef struct _xSidesGenericEscapeMessage
{

   USHORT Size;

   USHORT Code;

   ULONG SpecificFields[1];

} xSidesGenericEscapeMessage;
```

Input Messages

The formats of the messages that the xSides application sends to the video driver are described below.

IMSG_XSIDES_BEGIN

IMSG_XSIDES_BEGIN is sent to the video driver to indicate that the xSides application has begun execution and requires video driver support.

USHORT Code	USHORT Size

Code: **IMSG_XSIDES_BEGIN**

Size: 4

IMSG_XSIDES_END

IMSG_XSIDES_END is sent to the video driver to indicate that the xSides application is exiting, and no longer requires video driver support.

USHORT Code	USHORT Size

Code: **IMSG_XSIDES_END**

Size: 4

IMSG_XSIDES_QUERY_VIDEO_INFO

IMSG_XSIDES_QUERY_VIDEO_INFO is sent to the video driver to query information about the video hardware and driver, including the version and level of funtionality of xSides video driver extensions that the video

driver supports. The video driver returns the OMSG_XSIDES_VIDEO_INFO message.

USHORT Code	USHORT Size

Code: **IMSG_XSIDES_QUERY_VIDEO_INFO**

Size: 4

IMSG_XSIDES_CREATE_PIXELBAR

IMSG_XSIDES_CREATE_PIXELBAR is sent to the video driver to request the creation and display of a pixel bar on an edge of the display screen. The video driver returns the OMSG_XSIDES_STATUS message.

USHORT Code	USHORT Size
USHORT Edge	USHORT Extent
Flags	

Code: **IMSG_XSIDES_CREATE_PIXELBAR**

Size: 8

Edge: The edge of the screen to create the pixel bar on. May be XSIDES_EDGE_TOP, XSIDES_EDGE_BOTTOM, XSIDES_EDGE_LEFT, or XSIDES_EDGE_RIGHT, for the top, bottom, left and right edges, respectively. Only one pixel bar can be created per side; attempts to create further pixel bars on the same side will cause the video driver to return errors in the output message.

Extent: The number of scan lines high, or pixels wide that the pixel bar should be. Horizontal pixel bars (those on the top and bottom edges) span the entire current horizontal resolution of the display. Vertical pixel bars (those on the left and right edges) span the entire current vertical resolution of the display.

Flags: A field of flags that describe the properties of the pixel bar. Bit 0 is defined to be the active/inactive flag.

Bit 0: If 0, then the pixel bar is created in the inactive state, and is not visible, though memory for it is allocated from the video memory. If 1, then the pixel bar is created in the active state and space is created for it on the display surface.

IMSG_XSIDES_DELETE_PIXELBAR

IMSG_XSIDES_DELETE_PIXELBAR is sent to the video driver to delete a pixel bar previously created by an IMSG_XSIDES_CREATE_PIXELBAR message. The video driver returns the OMSG_XSIDES_STATUS message.

USHORT Code	USHORT Size
USHORT Edge	USHORT Reserved

Code: **IMSG_XSIDES_DELETE_PIXELBAR**

Size: 8

Edge: The edge of the screen that the pixel bar is on. May be XSIDES_EDGE_TOP, XSIDES_EDGE_BOTTOM, XSIDES_EDGE_LEFT, or XSIDES_EDGE_RIGHT, for the top, bottom, left and right edges, respectively. Attempts to delete a pixel bar that doesn't exist will cause the video driver to return error reports in the output message.

IMSG_XSIDES_ACTIVATE_PIXELBAR

IMSG_XSIDES_ACTIVATE_PIXELBAR is sent to the video driver to enable the display of a pixel bar previously created by an IMSG_XSIDES_CREATE_PIXELBAR message. The video driver returns the OMSG_XSIDES_STATUS message.

USHORT Code	USHORT Size
USHORT Edge	USHORT Reserved

Code: **IMSG_XSIDES_ACIVATE_PIXELBAR**

Size: 8

Edge: The edge of the screen that the pixel bar is on. May be XSIDES_EDGE_TOP, XSIDES_EDGE_BOTTOM, XSIDES_EDGE_LEFT, or XSIDES_EDGE_RIGHT, for the top, bottom, left and right edges, respectively. Attempts to activate a pixel bar that doesn't exist, or is already being displayed, will cause the video driver to return error reports in the output message.

IMSG_XSIDES_DEACTIVATE_PIXELBAR

IMSG_XSIDES_DEACTIVATE_PIXELBAR is sent to the video driver to disable the display of a pixel bar previously created by an IMSG_XSIDES_CREATE_PIXELBAR message. The video driver returns the OMSG_XSIDES_STATUS message.

USHORT Code	USHORT Size
USHORT Edge	USHORT Reserved

Code: **IMSG_XSIDES_DEACIVATE_PIXELBAR**

Size: 8

Edge: The edge of the screen that the pixel bar is on. May be XSIDES_EDGE_TOP, XSIDES_EDGE_BOTTOM, XSIDES_EDGE_LEFT, or XSIDES_EDGE_RIGHT, for the top, bottom, left and right edges, respectively. Attempts to deactivate a pixel bar that doesn't exist, or is already being inactive, will cause the video driver to return error reports in the output message.

IMSG_XSIDES_DISPLAY_PIXELBAR

IMSG_XSIDES_DISPLAY_PIXELBAR is sent to the video driver to display bitmaps on a pixel bar previously created by an IMSG_XSIDES_CREATE_PIXELBAR message. The video driver uses the SRCCOPY (0xCC) ROP to BitBlt the specified bitmap onto the pixel bar. The source bitmap address is specified in the lpszOutData field of the Escape function. The video driver returns the OMSG_XSIDES_STATUS message.

USHORT Code	USHORT Size
USHORT Edge	USHORT SrcStride
USHORT DestX	USHORT DestY
USHORT SrcX	USHORT SrcY
USHORT ExtX	USHORT ExtY

Code: **IMSG_XSIDES_DISPLAY_PIXELBAR**

Size: 20

Edge: The edge of the screen that the pixel bar is on. May be XSIDES_EDGE_TOP, XSIDES_EDGE_BOTTOM, XSIDES_EDGE_LEFT, or XSIDES_EDGE_RIGHT, for the top, bottom, left and right edges, respectively. Attempts to

display to a pixel bar that doesn't exist will cause the video driver to return an error.

SrcStride: The stride of the source bitmap. I.e. the number of bytes from the first pixel of a row to the first pixel of the next row in the bitmap.

DestX: The horizontal (x) location in the destination pixel bar, relative to the pixel bar origin (the pixel bar origin is at the upper left corner of the pixel bar, with x values increasing to the right and y values increasing towards the bottom.) where the source bitmap is to be written.

DestY: The vertical (y) location in the destination pixel bar, relative to the pixel bar origin (the pixel bar origin is at the upper left corner of the pixel bar, with x values increasing to the right and y values increasing towards the bottom.) where the source bitmap is to be written.

SrcX: The horizontal (x) location in the source bitmap, relative to the bitmap origin (the bitmap origin is at the upper left corner of the bitmap, with x values increasing to the right and y values increasing towards the bottom.) where the SRCCOPY BitBlt starts transferring data from.

SrcY: The vertical (y) location in the source bitmap, relative to the bitmap origin (the bitmap origin is at the upper left corner of the bitmap, with x values increasing to the right and y values increasing towards the bottom.) where the SRCCOPY BitBlt starts transferring data from.

IMSG_XSIDES_ENABLE_CURSOR

IMSG_XSIDES_ENABLE_CURSOR is sent to the video driver to enable the display of a mouse cursor icon a pixel bar created by an IMSG_XSIDES_CREATE_PIXELBAR message. The video driver returns the OMSG_XSIDES_STATUS message.

USHORT Code	USHORT Size
USHORT Edge	USHORT Reserved

Code: **IMSG_XSIDES_ENABLE_CURSOR**

Size: 8

Edge: The edge of the screen that the pixel bar is on. May be XSIDES_EDGE_TOP, XSIDES_EDGE_BOTTOM, XSIDES_EDGE_LEFT, or XSIDES_EDGE_RIGHT, for the top, bottom, left and right edges, respectively. Attempts to enable cursor display in pixel bar that doesn't exist, or where it is already enabled, will cause the video driver to return error reports in the output message.

IMSG_XSIDES_DISABLE_CURSOR

IMSG_XSIDES_DISABLE_CURSOR is sent to the video driver to enable the display of a mouse cursor icon a pixel bar created by an IMSG_XSIDES_CREATE_PIXELBAR message. The video driver returns the OMSG_XSIDES_STATUS message.

USHORT Code	USHORT Size
USHORT Edge	USHORT Reserved

Code: **IMSG_XSIDES_DISABLE_CURSOR**

Size: 8

Edge: The edge of the screen that the pixel bar is on. May be XSIDES_EDGE_TOP, XSIDES_EDGE_BOTTOM, XSIDES_EDGE_LEFT, or XSIDES_EDGE_RIGHT, for the top, bottom, left and right edges, respectively.

Note: Attempts to disable cursor display in pixel bar that doesn't exist, or where it is already disabled, will cause the video driver to return error reports in the output message.

IMSG_XSIDES_SET_CURSOR

IMSG_XSIDES_DISABLE_CURSOR is sent to the video driver to enable the display of a mouse cursor icon a pixel bar created by an IMSG_XSIDES_CREATE_PIXELBAR message. The video driver returns the OMSG_XSIDES_STATUS message. The cursor bitmap address is specified in the lpszOutData field of the Escape function.

USHORT Code	USHORT Size
USHORT Edge	USHORT Reserved

Code: **IMSG_XSIDES_DISABLE_CURSOR**

Size: 8

Edge: The edge of the screen that the pixel bar is on. May be XSIDES_EDGE_TOP, XSIDES_EDGE_BOTTOM, XSIDES_EDGE_LEFT, or XSIDES_EDGE_RIGHT, for the top, bottom, left and right edges, respectively.

IMSG_XSIDES_MOVE_CURSOR

IMSG_XSIDES_MOVE_CURSOR is sent to the video driver to position the cursor in a pixel bar created by an IMSG_XSIDES_CREATE_PIXELBAR message. The video driver returns the OMSG_XSIDES_STATUS message.

USHORT Code	USHORT Size
USHORT Edge	USHORT Reserved
USHORT DestX	USHORT DestY

Code: **IMSG_XSIDES_DISPLAY_PIXELBAR**

Size: 12

Edge: The edge of the screen that the pixel bar is on. May be XSIDES_EDGE_TOP, XSIDES_EDGE_BOTTOM, XSIDES_EDGE_LEFT, or XSIDES_EDGE_RIGHT, for the top, bottom, left and right edges, respectively. *DestX*: The horizontal (x) location in the destination pixel bar, relative to the pixel bar origin (the pixel bar origin is at the upper left corner of the pixel bar, with x values increasing to the right and y values increasing towards the bottom.) where the mouse cursor hotpoint is to be positioned.

DestY: The vertical (y) location in the destination pixel bar, relative to the pixel bar origin (the pixel bar origin is at the upper left corner of the pixel bar, with x values increasing to the right and y values increasing towards the bottom.) where the mouse cursor hotpoint it to be positioned.

Note: Attempts to position the cursor in a pixel bar that doesn't exist, or where the cursor isn't enabled, will cause the video driver to return an error.

Output Messages

OMSG_XSIDES_VIDEO_INFO

OMSG_XSIDES_VIDEO_INFO is returned by the video driver to the xSides Applications in response to an IMSG_XSIDES_QUERY_VIDEO_INFO message, and describes the version and level of xSides support provided as well as the video driver version and pertinent data about the video hardware.

USHORT Code	USHORT Size
USHORT	USHORT xSidesLevel

xSidesVersion	
VideoDriverVersion	
VideoHardware	
VideoConfigurationData	

Code: **OMSG_XSIDES_VIDEO_INFO**

Size: TBD

xSidesVersion: The version of xSides video driver extensions that the video driver supports.

xSidesLevel: The level of xSides functionality that the video driver supports. May be 0 (no support), 1 (bottom pixel bar only), 2 (bottom and top pixel bars) or 3 (bottom, top, left and right pixel bars.).

VideoDriverVersion: The name and version (release number) of the video driver and video miniport. This should be a much larger field, or set of fields, including character strings for file names. **Note: the format is TBD.**

VideoHardware: The names of the video hardware, including the board and chip, their manufacturers, versions and revisions, characteristics (e.g. amount of video RAM, the video bus interface, etc.). **Note: the format is TBD.**

VideoConfigurationData: The enumeration of the display resolutions supported, and the xSides resources available for each resolution, including specification of the number of extra pixels available and the resultant refresh frequency degradation. **Note: the format is TBD.**

OMSG_XSIDES_STATUS

OMSG_XSIDES_STATUS is returned by the video driver to the xSides application in response to various xSides input messages to report status and errors.

USHORT Code	USHORT Size
USHORT RequestingCode	USHORT RequestingSize
ULONG ErrorCode	

Code: **OMSG_XSIDES_STATUS**

Size: 12 (may be much larger.)

RequesitingCode: The message code of the request that caused the message to be returned.

RequestingSize: The size of the message whose request is causing the OMSG_XSIDES_STATUS to be returned.

ErrorCode: The status and error code being returned. This is TBD; more fields including test strings may be required.

Appendix A: GDI Escapes

GDI Escape Function:

Escape

The Escape function allows applications to access capabilities of a particular device not directly available through GDI. Escape calls made by an application are translated and sent to the display driver, to its DrvEscape function.

```
Escape (
 HDC        hdc,        // handle to device context
 int        nEscape,    //        function
 int        cbInput,    // number of bytes in input structure
 LPCSTR     lpvInData,  // pointer to input structure
 LPVOID     lpvOutData  // pointer to output structure
);
```

Parameters

hdc

Handle to the device context.

nEscape

Specifies the Escape function to be performed. This parameter must be one of the predefined Escape values. Use the **ExtEscape** function if your application defines a private Escape value.

cbInput

Specifies the number of bytes of data pointed to by the *lpvInData* parameter.

lpvInData

Pointer to the input structure required for the specified

lpvOutData

Pointer to the structure that receives output from this. This parameter should be NULL if no data is returned.

Return Values

If the function succeeds, the return value is greater than zero, except with the QUERYESCSUPPORT printer, which checks for implementation only. If the is not implemented, the return value is zero.

If the function fails, the return value is an error.

GDI ExtEscape Function

ExtEscape

The **ExtEscape** function allows applications to access capabilities of a particular device that are not available through GDI.

```
ExtEscape(
  HDC hdc,              // handle to device context
  int nEscape,          // escape function
  int cbInput,          // number of bytes in input
structure
  LPCSTR lpszInData,    // pointer to input structure
  int cbOutput,         // number of bytes in output
structure
  LPSTR lpszOutdata     // pointer to output
structure
);
```

Parameters

hdc

Handle to the device context.

nEscape

Specifies the escape function to be performed.

cbInput

Specifies the number of bytes of data pointed to by the *lpszInData* parameter.

lpszInData

Pointer to the input structure required for the specified escape.

cbOutput

Specifies the number of bytes of data pointed to by the *lpszOutData* parameter.

lpszOutData

Pointer to the structure that receives output from this escape. This parameter must not be NULL if **ExtEscape** is called as a query function. If no data is to be returned in this structure set *cbOutput* to 0.

Return Values

The return value specifies the outcome of the function. It is greater than zero if the function is successful, except for the QUERYESCSUPPORT printer escape, which checks for implementation only. The return value is zero if the escape is not implemented. A return value less than zero indicates an error.

Windows NT Video Driver DrvEscape Function:

DrvEscape

DrvEscape is used for retrieving information from a device that is not available in a device-independent device driver interface. The particular query depends on the value of the *iEsc* parameter. Drawing on the device is not allowed in this function. **DrvDrawEscape** is to be used for specialized drawing support.

```
ULONG DrvEscape(
    IN SURFOBJ*pso,
    IN ULONG iEsc,
    IN ULONG cjIn,
    IN PVOID *pvIn,
    IN ULONG cjOut,
    OUT PVOID *pvOut
);
```

Parameters

pso

Points to a SURFOBJ structure that describes the surface to which the call is directed.

iEsc

Specifies a query. The meaning of the other parameters depends on this value. ESC_QUERYESCSUPPORT is the only predefined value; it queries whether the driver supports a particular escape function. In this case, *pvIn* points to an escape function number; *cjOut* and *pvOut* are ignored. If the specified function is supported, the return value is nonzero.

cjIn

Specifies the size, in bytes, of the buffer pointed to by *pvIn*.

pvIn

Points to the input data for the call. The format of the input data depends on the query specified by the *iEsc* parameter.

cjOut

Specifies the size, in bytes, of the buffer pointed to by *pvOut*.

pvOut

Points to the output buffer. The format of the output data depends on the query specified by the *iEsc* parameter.

Return Value

The return value is dependent on the query specified by the *iEsc* parameter. If the function specified in the query is not supported, the return value is zero.

Appendix B: xSides Space

Creating the xSides Space

This section describes how the video driver can create the xSides Space beyond the Windows Desktop by reserving video memory contiguous to the video frame buffer and adjusting the CRTC parameters to increase the number of pixels that are displayed. Adjustments to the video timing registers can potentially be done to reduce the amount of frequency degradation caused by the increased screen surface.